What is change data capture (CDC)?

Article • 12/19/2023 • 26 contributors

Feedback

Applies to: SQL Server, Azure SQL Managed Instance

SQL Server & Azure SQL Managed Instance

In this article, learn about change data capture (CDC), which records activity on a database when tables and rows have been modified.

This article explains how CDC works with SQL Server and Azure SQL Managed Instance. For Azure SQL Database, see CDC with Azure SQL Database.

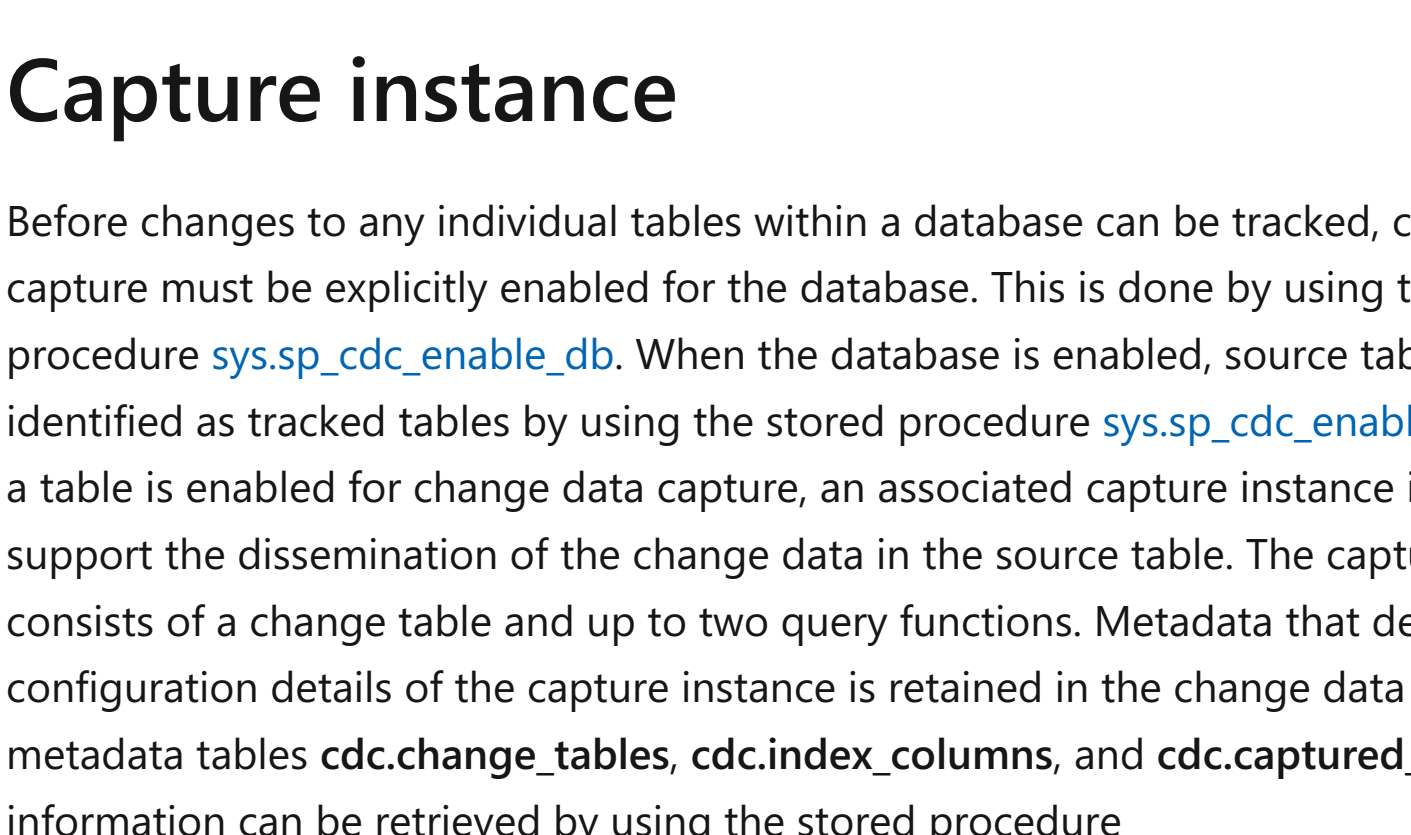
Overview

Change data capture utilizes the SQL Server Agent to log insertions, updates, and deletions occurring in a table. So, it makes these data changes accessible to be easily consumed using a relational format. The column data and essential metadata need to apply these change data to a target environment are captured for the modified rows and stored in change tables that mirror the column structure of the tracked source tables. Furthermore, table-valued functions are available for systematic access to this change data by consumers.

A good example of a data consumer this technology targets is an extraction, transformation, and loading (ETL) application. An ETL application incrementally loads change data from SQL Server source tables to a data warehouse or data mart. Although the representation of the source tables within the data warehouse must reflect changes in the source tables, an end-to-end technology that refreshes a replica of the source isn't appropriate. Instead, you need a reliable stream of change data that is structured so that consumers can apply it to dissimilar target representations of the data. SQL Server change data capture provides this technology.

Data flow

The following illustration shows the principal data flow for change data capture.



The source of change data for change data capture is the SQL Server transaction log. As inserts, updates, and deletes are applied to tracked source tables, entries that describe those changes are added to the log. The log serves as input to the capture process. Then, it reads the log and adds information about changes to the tracked table's associated change table. Functions are provided to enumerate the changes that appear in the change tables over a specified range, returning the information in the form of a filtered result set. The filtered result set is typically used by an application process to update a representation of the source in some external environment.

Capture instance

Before changes to any individual tables within a database can be tracked, change data capture must be explicitly enabled for the database. This is done by using the stored procedure sys.sp_cdc_enable_db. When the database is enabled, source tables can be identified as tracked tables by using the stored procedure sys.sp_cdc_enable_table. When a table is enabled for change data capture, an associated capture instance is created to support the dissemination of the change data in the source table. The capture instance consists of a change table and up to two query functions. Metadata that describes the configuration details of the capture instance is retained in the change data capture metadata tables **cdc.change_tables**, **cdc.index_columns**, and **cdc.captured_columns**. This information can be retrieved by using the stored procedure sys.sp_cdc_help_change_data_capture.

All objects that are associated with a capture instance are created in the change data capture schema of the enabled database. The requirements for the capture instance name are that it's a valid object name, and that it's unique across the database capture instances. By default, the name is <*schema name_table name*> of the source table. Its associated change table is named by appending **_CT** to the capture instance name. The function that is used to query for all changes is named by prepending **fn_cdc_get_all_changes_** to the capture instance name. If the capture instance is configured to support **net changes**, the **net_changes** query function is also created and named by prepending **fn_cdc_get_net_changes_** to the capture instance name.

> **Important**
>
> The maximum number of capture instances that can be concurrently associated with a single source table is two.

Change table

The first five columns of a change data capture change table are metadata columns. These provide additional information that is relevant to the recorded change. The remaining columns mirror the identified captured columns from the source table in name and, typically, in type. These columns hold the captured column data that is gathered from the source table.

Each insert or delete operation that is applied to a source table appears as a single row within the change table. The data columns of the row that results from an insert operation contain the column values after the insert. The data columns of the row that results from a delete operation contain the column values before the delete. An update operation requires one-row entry to identify the column values before the update, and a second row entry to identify the column values after the update.

Each row in a change table also contains other metadata to allow interpretation of the change activity. The column __$start_lsn identifies the commit log sequence number (LSN) that was assigned to the change. The commit LSN both identifies changes that were committed within the same transaction, and orders those transactions. The column __$seqval can be used to order more changes that occur in the same transaction. The column __$operation records the operation that is associated with the change: 1 = delete, 2 = insert, 3 = update (before image), and 4 = update (after image). The column __$update_mask is a variable bit mask with one defined bit for each captured column. For insert and, delete entries, the update mask has all bits set. Update rows, however, will have those bits set that corresponds to changed columns.

Validity interval

The change data capture validity interval for a database is the time during which change data is available for capture instances. The validity interval begins when the first capture instance is created for a database table, and continues to the present time.

Database

Data that is deposited in change tables grow unmanageably if you don't periodically and systematically prune the data. The change data capture cleanup process is responsible for enforcing the retention-based cleanup policy. First, it moves the low endpoint of the validity interval to satisfy the time restriction. Then, it removes expired change table entries. By default, three days of data are retained.

At the high end, as the capture process commits each new batch of change data, new entries are added to **cdc.lsn_time_mapping** for each transaction that has change table entries. Within the mapping table, both a commit Log Sequence Number (LSN) and a transaction commit time (columns start_lsn and tran_end_time, respectively) are retained. The maximum LSN value that is found in **cdc.lsn_time_mapping** represents the high water mark of the database validity window. Its corresponding commit time is used as the base from which retention-based cleanup computes a new low water mark.

Because the capture process extracts change data from the transaction log, there's a built-in latency between the time that a change is committed to a source table and the time that the change appears within its associated change table. While this latency is typically small, it's nevertheless important to remember that change data isn't available until the capture process has processed the related log entries.

Capture instance

Although it's common for the database validity interval and the validity interval of individual capture instance to coincide, however, isn't always true. The validity interval of the capture instance starts when the capture process recognizes the capture instance and starts to log associated changes to its change table. As a result, if capture instances are created at different times, each will have a different low endpoint. The start_lsn column of the result set that is returned by sys.sp_cdc_help_change_data_capture shows the current low endpoint for each defined capture instance. When the cleanup process cleans up change table entries, it adjusts the start_lsn values for all capture instances to reflect the new low water mark for available change data. Only those capture instances that have start_lsn values that are currently less than the new low water mark are adjusted. Over time, if no new capture instances are created, the validity intervals for all individual instances will tend to coincide with the database validity interval.

The validity interval is important to consumers of change data because the extraction interval for a request must be fully covered by the current change data capture validity interval for the capture instance. If the low endpoint of the extraction interval is to the left of the low endpoint of the validity interval, there could be missing change data due to aggressive cleanup. If the high endpoint of the extraction interval is to the right of the high endpoint of the validity interval, it indicates that the capture process hasn't yet been processed through the time represented by the extraction interval, and there could also be missing change data.

The function sys.fn_cdc_get_min_lsn is used to retrieve the current minimum LSN for a capture instance, while sys.fn_cdc_get_max_lsn is used to retrieve the current maximum LSN value. When you query the change data, if the specified LSN range doesn't lie within these two LSN values, the change data capture query functions fail.

Handling changes to source table

Accommodating column changes in the source tables that are being tracked is a difficult issue for downstream consumers. Although enabling change data capture on a source table doesn't prevent such DDL changes from occurring, change data capture mitigates the effect on consumers by preserving the delivered result sets returned through the API, even as the column structure of the underlying source table changes. This fixed column structure is also reflected in the underlying change table that the defined query functions access.

The capture process responsible for populating the change table accommodates a fixed column structure change table by ignoring any new columns not identified for capture when the source table was enabled for change data capture. If a tracked column is dropped, null values are supplied for the column in the subsequent change entries. However, if an existing column undergoes a change in its data type, the change is propagated to the change table to ensure that the capture mechanism doesn't introduce data loss to tracked columns. The capture process also posts any detected changes to the column structure of tracked tables to the cdc.ddl_history table. Consumers wishing to be alerted of adjustments that might have to be made in downstream applications, use the stored procedure sys.sp_cdc_get_ddl_history.

Typically, the current capture instance continues to retain its shape when DDL changes are applied to its associated source table. However, it's possible to create a second capture instance for the table that reflects the new column structure. This option allows the capture process to make changes to the same source table into two distinct change tables having two different column structures. Thus, while one change table can continue to feed current operational programs, the second one can drive a development environment that is trying to incorporate the new column data. Allowing the capture mechanism to populate both change tables in tandem means that a transition from one to the other can be accomplished without loss of change data. This can happen anytime the two change data capture timelines overlap. When the transition is affected, the obsolete capture instance can be removed.

> **Important**
>
> The maximum number of capture instances that can be concurrently associated with a single source table is two.

Relationship with log reader agent

The logic for change data capture process is embedded in the stored procedure sp_replcmds, an internal server function built as part of sqlservr.exe and also used by transactional replication to harvest changes from the transaction log. In SQL Server and Azure SQL Managed Instance, when change data capture alone is enabled for a database, you create the change data capture SQL Server Agent capture job as the vehicle for invoking sp_replcmds. When replication is also present, the transactional log reader alone is used to satisfy the change data needs for both of these consumers. This strategy significantly reduces log contention when both replication and change data capture are enabled for the same database.

The switch between these two operational modes for capturing change data occurs automatically whenever there's a change in the replication status of a change data capture enabled database.

> **Note**
>
> In SQL Server and Azure SQL Managed Instance, both instances of the capture logic require SQL Server Agent to be running for the process to execute.

The principal task of the capture process is to scan the log and write column data and transaction-related information to the change data capture change tables. To ensure a transactionally consistent boundary across all the change data capture change tables that it populates, the capture process opens and commits its own transaction on each scan cycle. It detects when tables are newly enabled for change data capture, and automatically includes them in the set of tables that are actively monitored for change entries in the log. Similarly, disabling change data capture will also be detected, causing the source table to be removed from the set of tables actively monitored for change data. When processing for a section of the log is finished, the capture process signals the server log truncation logic, which uses this information to identify log entries eligible for truncation.

> **Important**
>
> When a database is enabled for change data capture, even if the recovery mode is set to simple recovery the log truncation point will not advance until all the changes that are marked for capture have been gathered by the capture process. If the capture process is not running and there are changes to be gathered, executing CHECKPOINT will not truncate the log.

The capture process is also used to maintain history on the DDL changes to tracked tables. The DDL statements that are associated with change data capture make entries to the database transaction log whenever a change data capture-enabled database or table is dropped or columns of a change data capture-enabled table are added, modified, or dropped. These log entries are processed by the capture process, which then posts the associated DDL events to the cdc.ddl_history table. You can obtain information about DDL events that affect tracked tables by using the stored procedure sys.sp_cdc_get_ddl_history.

> **Warning**
>
> - MaxCmdsInTran was not designed to always be turned on. It exists to work around cases where someone accidentally performed a large number of DML operations in a single transaction (causing a delay in the distribution of commands until the entire transaction is in the distribution database, locks being held, etc.). If you routinely fall into this situation, review your application logic to find ways to reduce the transaction size.
> - MaxCmdsInTran is not supported if the given publication database has both CDC and replication enabled. Using **MaxCmdsInTran** in this configuration may lead to data loss in CDC change tables. It may also cause PK errors if the **MaxCmdsInTran** parameter is added and removed while replicating a large Transaction.

Agent jobs

Two SQL Server Agent jobs are typically associated with a change data capture enabled database: one that is used to populate the database change tables, and one that is responsible for change table cleanup. Both jobs consist of a single step that runs a Transact-SQL command. The Transact-SQL command that is invoked is a change data capture defined stored procedure that implements the logic of the job. The jobs are created when the first table of the database is enabled for change data capture. The Cleanup Job is always created. The capture job will only be created if there are no defined transactional publications for the database. The capture job is also created when both change data capture and transactional replication are enabled for a database, and the transactional log reader job is removed because the database no longer has defined publications.

Both the capture and cleanup jobs are created by using default parameters. The capture job is started immediately. It runs continuously, processing a maximum of 1000 transactions per scan cycle with a wait of 5 seconds between cycles. The cleanup job runs daily at 2 A.M. It retains change table entries for 4320 minutes or 3 days, removing a maximum of 5000 entries with a single delete statement.

The change data capture agent jobs are removed when change data capture is disabled for a database. The capture job can also be removed when the first publication is added to a database, and both change data capture and transactional replication are enabled.

Internally, change data capture agent jobs are created and dropped by using the stored procedures sys.sp_cdc_add_job and sys.sp_cdc_drop_job, respectively. These stored procedures are also exposed so that administrators can control the creation and removal of these jobs.

An administrator has no explicit control over the default configuration of the change data capture agent jobs. The stored procedure sys.sp_cdc_change_job is provided to allow the default configuration parameters to be modified. In addition, the stored procedure sys.sp_cdc_help_jobs allows current configuration parameters to be viewed. Both the capture job and the cleanup job extract configuration parameters from the table msdb.dbo.cdc_jobs on startup. Any changes made to these values by using sys.sp_cdc_change_job won't take effect until the job is stopped and restarted.

Two other stored procedures are provided to allow the change data capture agent jobs to be started and stopped: sys.sp_cdc_start_job and sys.sp_cdc_stop_job.

> **Note**
>
> Starting and stopping the capture job does not result in a loss of change data. It only prevents the capture process from actively scanning the log for change entries to deposit in the change tables. A reasonable strategy to prevent log scanning from adding load during periods of peak demand is to stop the capture job and restart it when demand is reduced.

Both SQL Server Agent jobs were designed to be flexible enough and sufficiently configurable to meet the basic needs of change data capture environments. In both cases, however, the underlying stored procedures that provide the core functionality have been exposed so that further customization is possible.

Change data capture can't function properly when the Database Engine service or the SQL Server Agent service is running under the NETWORK SERVICE account. This can result in error 22832.

Interoperability with other features

Change data capture has some limitations when working with other SQL Server features. Review Interoperability to learn more.

Known issues

For known issues and errors associated with change data capture, review Known issues with CDC.

See also

- Known issues and limitations
- Work with Change Data
- Track Data Changes
- Enable and Disable change data capture
- Administer and Monitor change data capture
- Temporal Tables

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Additional resources

Documentation

Create a Python function from the command line - Azure Functions
Learn how to create a Python function from the command line, then publish the local project to serverless hosting in Azure Functions.

Enable and Disable change data capture - SQL Server
Enable and Disable change data capture

Python developer reference for Azure Functions
Understand how to develop, validate, and deploy your Python code projects to Azure Functions using the Python library for Azure Functions.

Show 4 more

Training

Module
Data tracking and synchronization with Azure SQL Database - Training
Azure SQL data tracking module that covers data changes tracking. The module explores tools such as change data capture (CDC) and change tracking.

Certification
Microsoft Certified: Azure Database Administrator Associate - Certifications
Administer an SQL Server database infrastructure for cloud, on-premises and hybrid relational databases using the Microsoft PaaS relational database offerings.

Events

Join AI Skills Fest Challenge
Apr 8 at 10 AM - May 28 at 2 AM
Sharpen your AI skills and enter the sweepstakes to win a free Certification exam
Register now!